No Free Lunch for Branch and Bound

Edmund Burke, Jerry Swan, John Woodward

http://www.cs.stir.ac.uk/~jrw/

jrw@cs.stir.ac.uk

University of Stirling.

Abstract

While **meta-heuristics** are being increasingly adopted by operational research practitioners for problems which are intractable, the ``**No Free Lunch'' theorems** state that over all **black-box optimization problems**, all meta-heuristics have indistinguishable performance.

Branch and bound is an operational research algorithm which **prunes the search space by discarding candidate solutions**. However, branch and bound was explicitly excluded from the original No Free Lunch theorems as it **makes use of domain knowledge** which is not exploited by black-box meta-heuristics. In this paper we prove that variants of branch and bound are indeed subject to ``No Free Lunch''.

The Talk In a Sound Bite

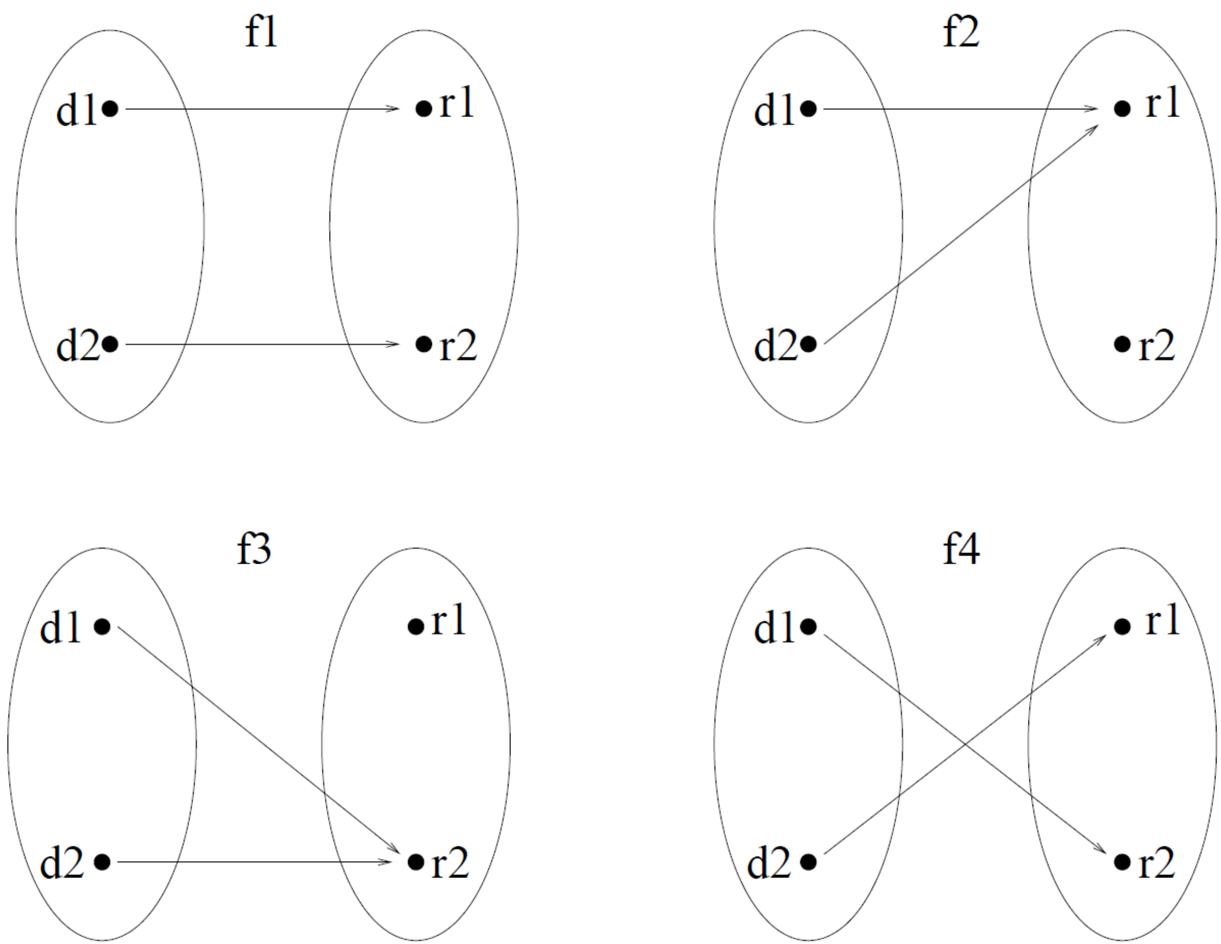
- Metaheuristics can be used in conjunction with “classical algorithms” (exact) e.g. branch and bound. “matheuristics”
- Exact algorithms make use of domain knowledge e.g. objective function is positive and therefore we can estimate bounds.
- Over all problems no metaheuristic is better than any other when used in conjunction with branch and bound. (deeper NFL)

Outline of Talk

- Consider searching “all functions” (small e.g.)
- Simple (intuitive) proof of No Free Lunch.
- Branch and Bound (with a small example)
- Definitions (metaheuristic, performance, …)
- Statement of NFL.
- Proof for Branch and Bound.
- Conclusions.

All Functions (2^2)

given f(d1) does this help estimate f(d2) ???

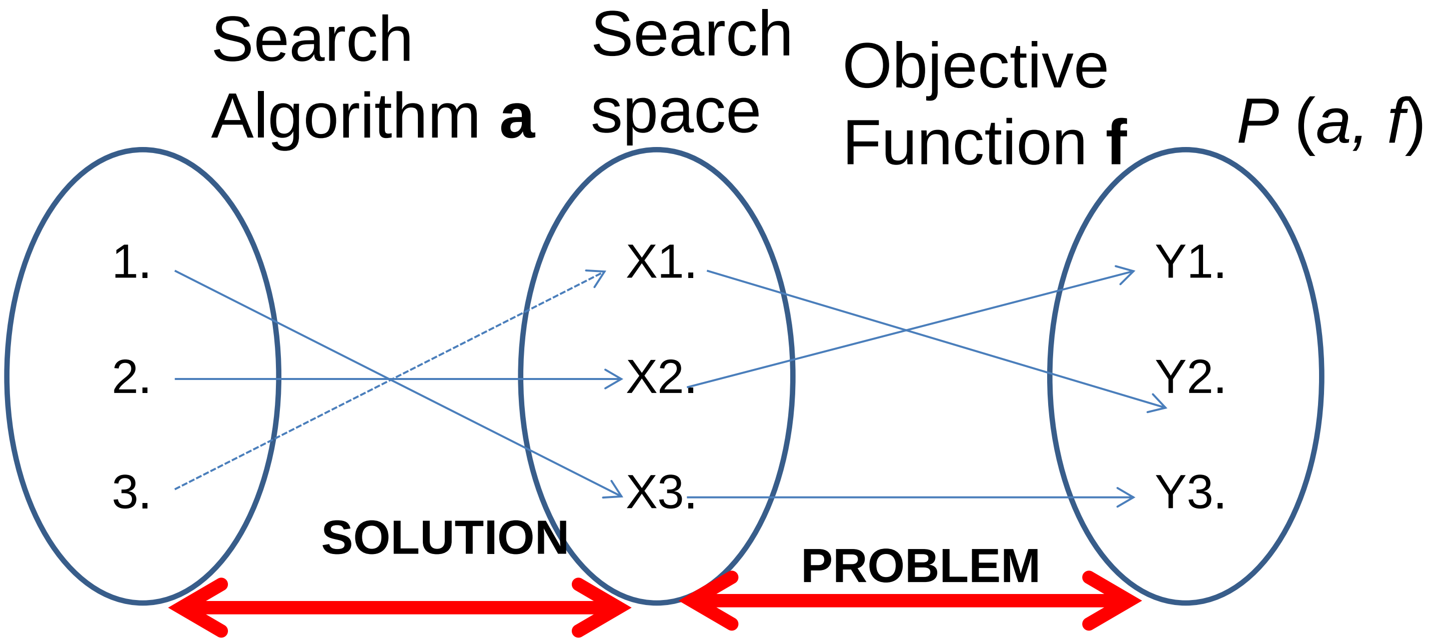


Definitions

- A **metaheuristic** is a list of points in the search space <x1, x2, …, >
- A **function** is a list (look up table) of points in the range of the function <y1, y2, …, >
- A **performance vector** results from of applying a metaheuristic to a function <y1, y2, …, >
- A **performance measure** is a function of the performance vector. P(<y1, y1, …, >)

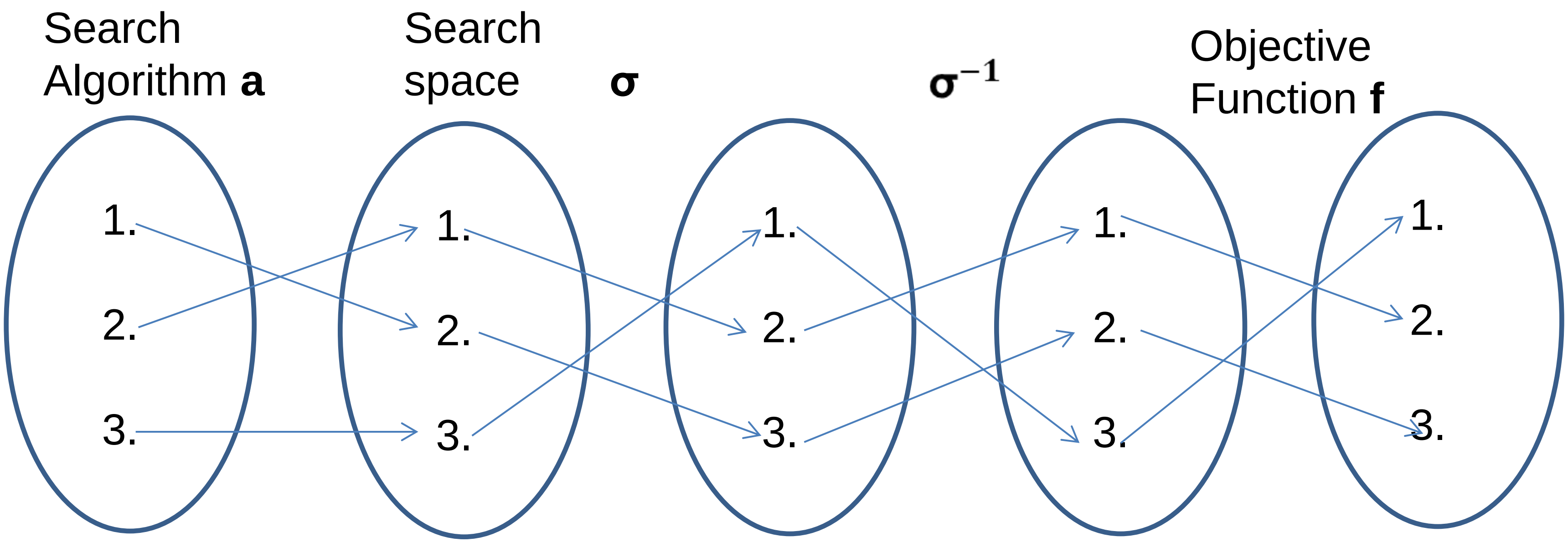
Which cup is the pea under (see recommended paper later - metaphor)

Theoretical Motivation 1



1. A **search space** contains the <u>set of all possible solutions</u>.
2. An **objective function** determines the <u>quality of solution</u>.
3. A **search algorithm** determines the <u>sampling order</u> (i.e. enumerates i.e. without replacement). It is a (approximate) permutation.
4. **Performance measure** *P* (*a*, *f*) depend only on y1, y2, y3
5. **<u>Aim find a solution with a near-optimal objective value using a search algorithm.</u>** ANY QUESTIONS BEFORE NEXT SLIDE?

Theoretical Motivation 2



$P\,(a, f) = P\,(a\,\boldsymbol{\sigma}, \boldsymbol{\sigma}^{-1} f)$ $\qquad P\,(A, F) = P\,(A\boldsymbol{\sigma}, \boldsymbol{\sigma}^{-1} F)$

P is a **performance measure**, (<u>based only on output values</u>).

A and F are probability distributions over algorithms and functions). **F is a problem class.** ASSUMPTIONS IMPLICATIONS

1. Algorithm **a** applied to function $\boldsymbol{\sigma\sigma}^{-1}\boldsymbol{f}$ (that is $\boldsymbol{f}$)
2. Algorithm **a**$\boldsymbol{\sigma}$ applied to function $\boldsymbol{\sigma}^{-1}\boldsymbol{f}$ precisely identical.

All Functions (2^3)

given f(x0) and f(x1), does this help estimate f(x2) ???

	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
x_0	y_0	y_1	y_0	y_1	y_0	y_1	y_0	y_1
x_1	y_0	y_0	y_1	y_1	y_0	y_0	y_1	y_1
x_2	y_0	y_0	y_0	y_0	y_1	y_1	y_1	y_1

Machine Learning.

We cannot extrapolate/generalize from the training set to the test set (???).

p(f)=**p(c|e)**, given example e, we want to predict which class c it belongs too. This is equivalent to known the distribution over the set of functions.

	Inputs			f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	...
	0	0	0	0	0	0	0	0	0	0	0	0	0	...
	0	0	1	0	0	0	0	0	0	0	0	0	0	...
Training	0	1	0	0	0	0	0	0	0	0	0	0	0	...
Set	0	1	1	0	0	0	0	0	0	0	0	0	0	...
	1	0	0	0	0	0	0	0	0	0	0	1	1	...
	1	0	1	0	0	0	0	1	1	1	1	0	0	...
Test	1	1	0	0	0	1	1	0	0	1	1	0	0	...
Set	1	1	1	0	1	0	1	0	1	0	1	0	1	...

Branch and Bound Algorithm

1. an "enumeration" of all candidate solutions, solutions are built incrementally.
2. subsets of candidates can be discarded, using upper and lower bounds.
3. This requires some knowledge of how the objective function behaves (properties).
4. Example are TSP and knapsack – we know when to "bail-out" of a poor set of solution and effectively discard that subset.

Person-Task Problem

1. 4 people {A,B,C,D} and 4 tasks {1,2,3,4}
2. The table below shows the number of minutes for each person to complete each task.
3. Each person does one task.
4. Each task needs an assigned a person.
5. Minimize total time taken.
6. How do we assign people to tasks?
7. E.g. ACDB =9+1+2+6 = 18 minutes in total.

	Task 1	Task 2	Task 3	Task 4
Person A	9	5	4	5
Person B	4	3	5	6
Person C	3	1	3	2
Person D	2	4	2	6

Example of Bounding Function

1. Example, calculate best value given partial assignment A???
2. (A does task 1, other tasks are not yet assigned)?
3. The cost of assigning person A to task 1 is 9 minutes
4. Best <u>*unassigned*</u> person for
5. 2 is C (1), 3 is D (2), 4 is C (2) in (minutes)
6. Note that C is assigned twice!
7. Total time = (9+1+2+2) = 14 minutes.
8. We want to minimize so the bounding function is an underestimate.

Branch and Bound Stage 1

Task 0

Task 1

Task 2

Task 3

Incumbent (best complete) solution

None yet.

	1	2	3	4
A	9	5	4	5
B	4	3	5	6
C	3	1	3	2
D	2	4	2	6

PRUNED

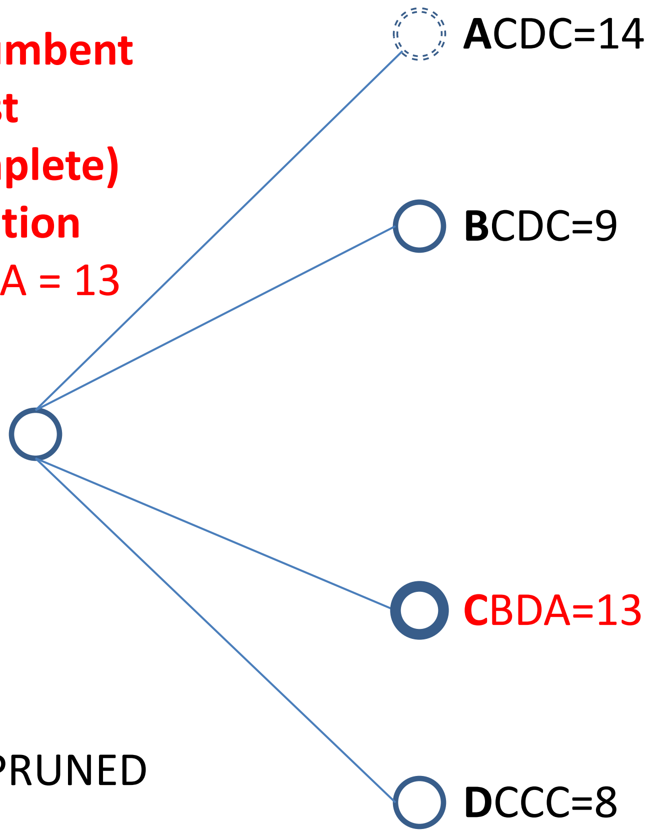
FEASIBLE

PRUNED & FESIBLE

Branch and Bound Stage 2

Task 0 | Task 1 | Task 2 | Task 3

Incumbent (best complete) solution
CBDA = 13



Pruned as > 13

promising

Feasible so is1st incumbent. There is no point growing this node any more.

promising

	1	2	3	4
A	9	5	4	5
B	4	3	5	6
C	3	1	3	2
D	2	4	2	6

PRUNED

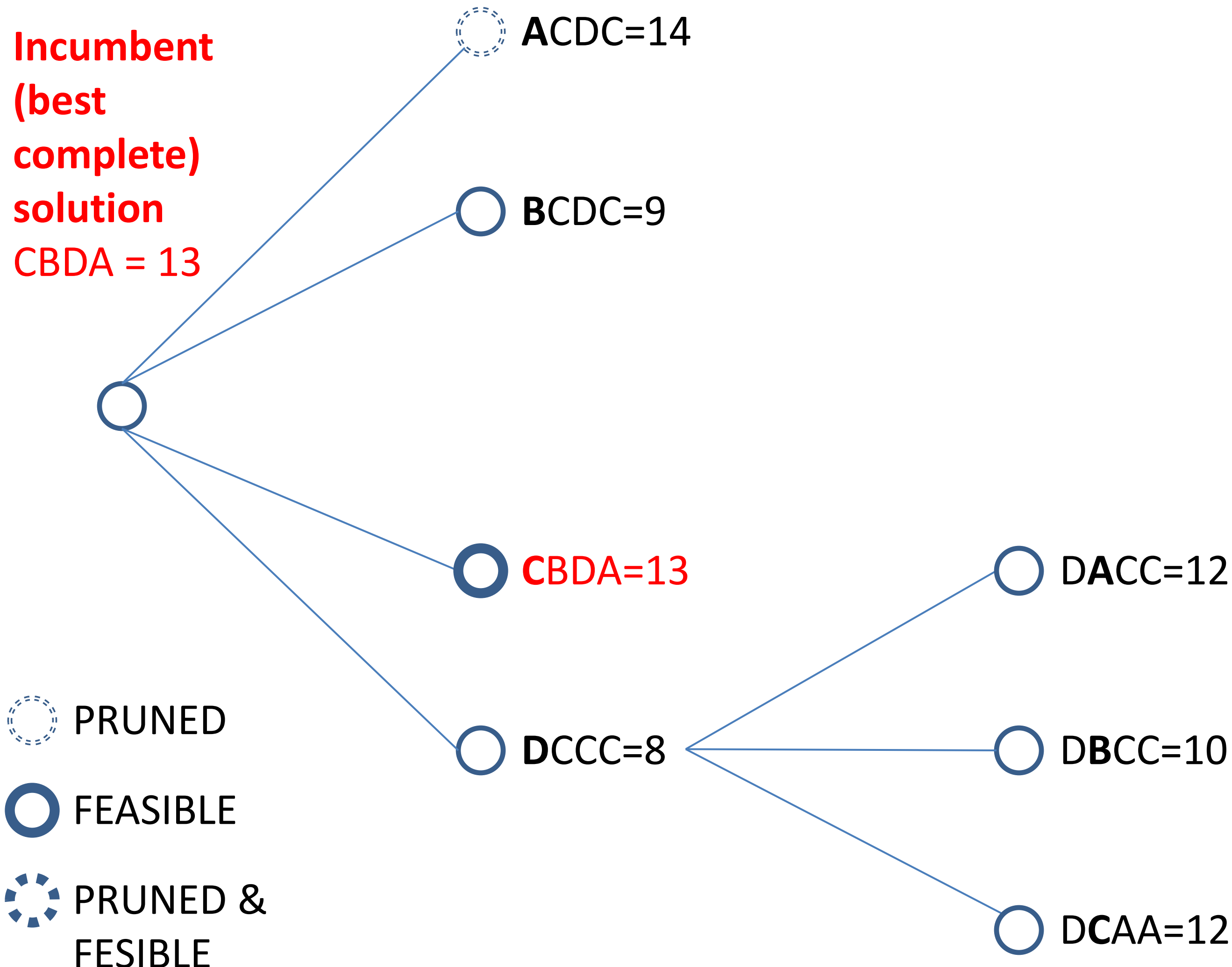
FEASIBLE

PRUNED & FESIBLE

Branch and Bound Stage 3

Task 0 | Task 1 | Task 2 | Task 3

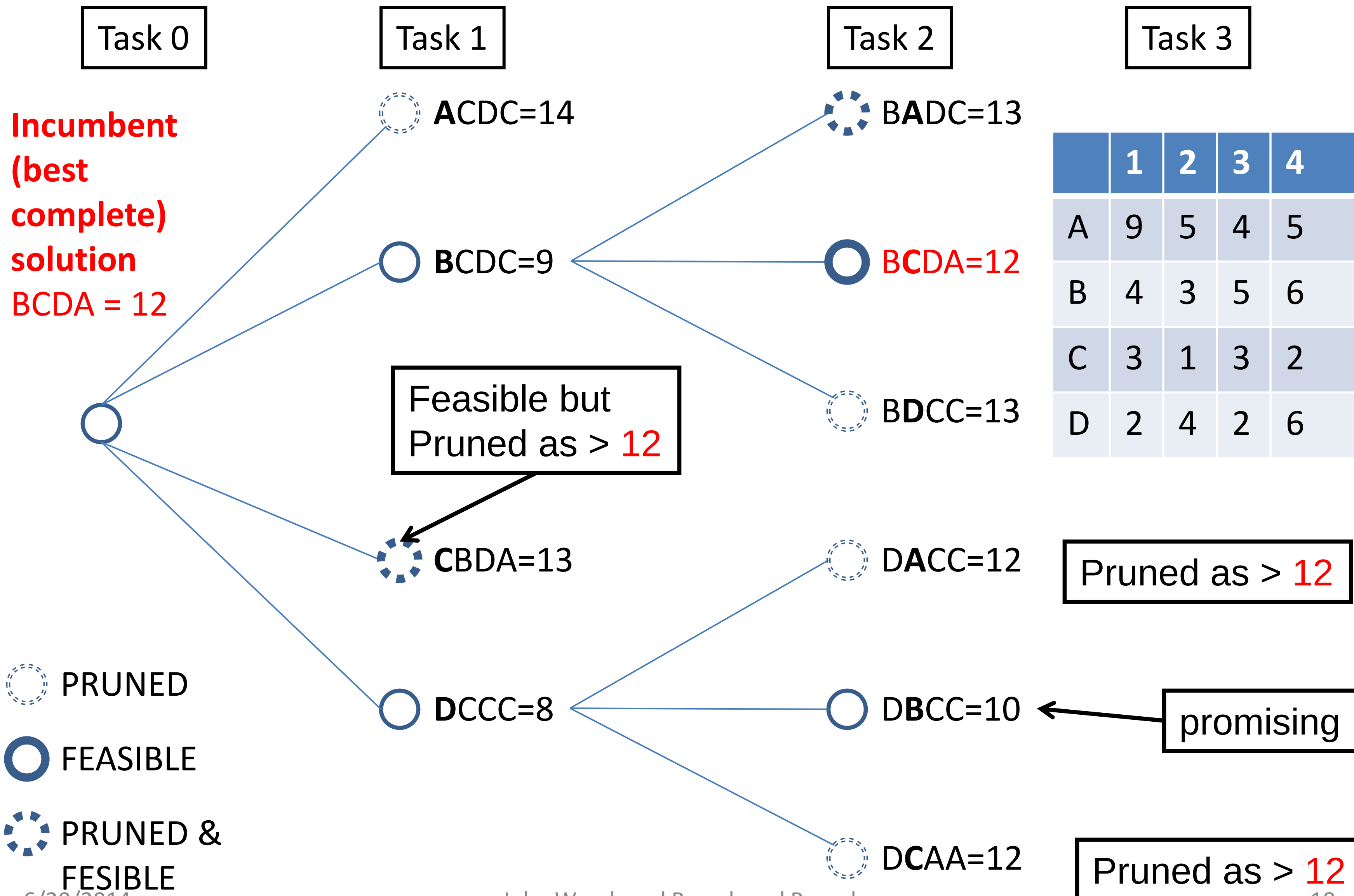
Incumbent (best complete) solution
CBDA = 13



	1	2	3	4
A	9	5	4	5
B	4	3	5	6
C	3	1	3	2
D	2	4	2	6

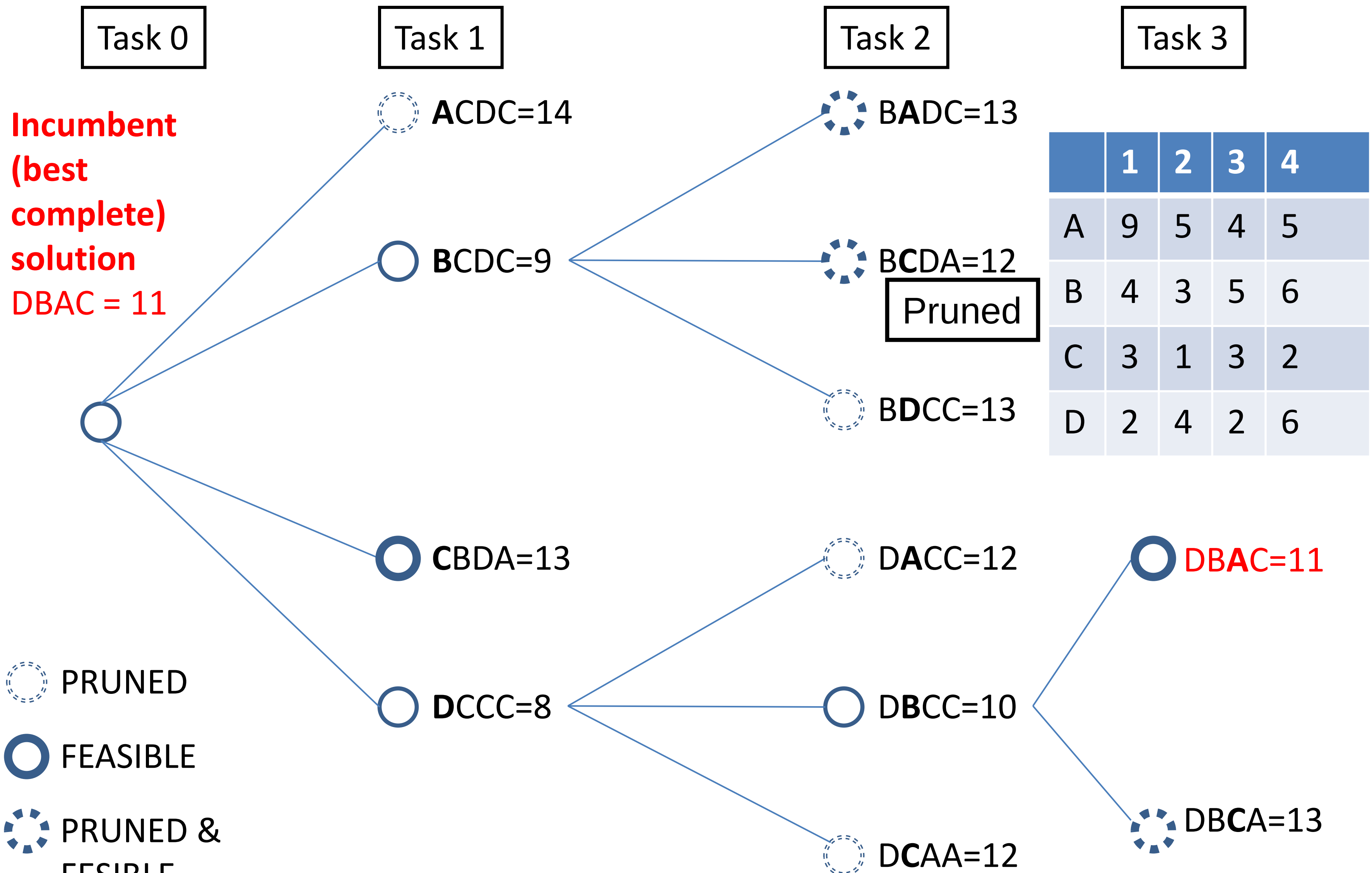
No new feasible solutions, therefore no new incumbent

Branch and Bound Stage 4



	1	2	3	4
A	9	5	4	5
B	4	3	5	6
C	3	1	3	2
D	2	4	2	6

Branch and Bound Stage 5



	1	2	3	4
A	9	5	4	5
B	4	3	5	6
C	3	1	3	2
D	2	4	2	6

Search Space of Complete Solutions

ABCD
BACD
….
DCBA

A, B, C, D
AB, AC, …
BA, BC, …
….
DCBA

- If we consider complete solutions (`whole` permutations).
- What about if we use branch and bound?
- If we consider partial solutions…

NFL Theorem

- Over the set of all **functions**, and two **metaheuristics** generate precisely the same collection of **performance vectors**.
- Consider 2 metaheuristics and 4 functions

	$f_{<y_1,y_2>}$	$f_{<y_1,y_1>}$	$f_{<y_2,y_2>}$	$f_{<y_2,y_1>}$
$< x_1, x_2 >$	$< y_1, y_2 >$	$< y_1, y_1 >$	$< y_2, y_2 >$	$< y_2, y_1 >$
$< x_2, x_1 >$	$< y_2, y_1 >$	$< y_1, y_1 >$	$< y_2, y_2 >$	$< y_1, y_2 >$

NO FREE LUNCH AND BRANCH AND BOUND

- NFL considers evaluations of the objective function with “**complete solutions**”. <x1, x2, …, >
- We need to extend the search space to include all “**partial solutions”**. (person1,person2) <(p1),(p2),…,(p1,p2),…,(p4,p3,p2,p1)>
- But the space of partial solutions, can be enumerated and considered as a space in its own right. Therefore the original theorem holds over this space. $P\ (a,\ f) = P\ (a\ \boldsymbol{\sigma}.\boldsymbol{\sigma}^{-1}\ f)$

Illustration

- In the case of the person-task assignment problem, we could expand the least-cost partial solutions first.
- However this "greedy" approach may not be optimal.
- $objective = \sum fi$ (fi>0) We could permute the fi
- We have argued that, over all objective functions, no one metaheuristic outperforms any other.

Must read papers

Toward a Justification of Meta-learning: Is the No Free Lunch Theorem a Show-stopper?

Christophe Giraud-Carrier.

Metaheuristics—the metaphor exposed

Kenneth Sörensen

Other Papers

Unbiased Black Box Search Algorithms

Jonathan E. Rowe Michael D. Vose

Edgar A. Duéñez-Guzmán, Michael D. Vose: **No Free Lunch and Benchmarks.** Evolutionary Computation 21(2): 293-312 (2013)

My own papers

The Necessity of Meta Bias in Search Algorithms. International Conference on Computational Intelligence and Software Engineering 2010.

Computable and Incomputable Search Algorithms and Functions. IEEE International Conference on Intelligent Computing and Intelligent Systems (IEEE ICIS 2009)

GA or GP, that is not the question, Congress on Evolutionary Computation 2003

No Free Lunch, Program Induction and Combinatorial Problems, EuroGP 2003 Essex, UK

Conclusions

- An algorithm which makes use of domain knowledge and can be used to discard/prune parts of the search space will (on average) perform better that algorithms that do not. (**obvious**).
- **If we take an algorithm which makes use of domain knowledge and supplement it with a two different metaheuristics, neither does better on average.**
- Automatic design of Matheuristics are an obvious direction.

The End

- Thank you for you attention.
- Any questions.
- 7 fully funded PhD positions at Stirling!!!
- http://www.cs.stir.ac.uk/~jrw/
- jrw@cs.stir.ac.uk
- Workshop at GECCO on automatic design of algorithms.